2019/10/28, 2019-2 오픈소스리눅스 프로그래밍

중간고사 문제 풀이

최서윤 (seoyun@pel.smuc.ac.kr)

상명대학교 프로토콜공학연구실

목 차

- github 업로드
- Q1
  - 두 개의 실수를 입력받아 평균 출력하기
- Q2
  - 1부터 1000까지 한 줄에 10개씩 출력하기
- Q3
  - swap함수와 포인터를 사용해 두 변수 값 교환하기
- Q4
  - 팩토리얼 계산 프로그램 오류 GDB로 해결하기

github 업로드

1. github에 oslp 중간고사 전용 레파지토리 생성 후 주소 복사
2. cd (작업한 디렉토리)
3. git init
4. git config --global user.email “(가입한 메일주소)”
5. git config --global user.name “(가입한 유저네임)”
6. git add .
7. git commit –m “(작업에 대한 설명)”
8. git remote add (origin) (아까 복사한 레파지토리 주소)
9. git push (origin) master

이미 origin이 있다고 뜰 경우 다른 이름으로 할 것

Q1

- 문제

1. 2개의 실수를 입력 받아 평균을 출력하는 프로그램을 작성하시오.

    - 정의해야 하는 함수: *double average(double in1, double in2)*

Q1

- 문제 분석!!

printf()와 scanf()로 2개의 실수를 입력받아야겠네!

최종 목표는 입력받은 두 실수의 평균을 출력하는 것!!

1. 2개의 실수를 입력 받아 평균을 출력하는 프로그램을 작성하시오.

- 정의해야 하는 함수: *double average(double in1, double in2)*

사용자 정의 함수를 만들어서 평균을 구해라...!!

double형의 변수를 main에 반환해야하는군...!

Q1

- 사용자 정의 함수?
  - main 함수 외에 사용자가 직접 제작한 함수
    - argument(전달인자)

```
int main() {
        int a, b;
        int result;              // result는 아규먼트 answer을 위한 파라미터
        result=sum(a,b);         // 아규먼트 a, 아규먼트 b, sum 함수 호출
}
```

    - parameter (매개변수)

```
int sum (int par1, int par2) {        // par1은 아규먼트a를 위한 파라미터
        int answer;                      par2는 아규먼트 b를 위한 파라미터
        answer = par1+par2
        return(answer);               //answer은 반환값
}
```

Q1

- 사용자 정의 함수?
  - 사용법

- main 함수 전에 정의 및 선언 되어야해요!

```
#incldue<stdio.h>
double calculate (int 파라미터1, int 파라미터2);
```

(main 함수 전에 선언할 때는 꼭 세미콜론을 붙여주세요!)

```
int main() {
        …
        result = calculate (아규먼트1, 아규먼트2)
}
```

- main함수에 반환할 데이터 타입

```
double calculate (int 파라미터1, int 파라미터2) {
        …
        return(result);
}
```

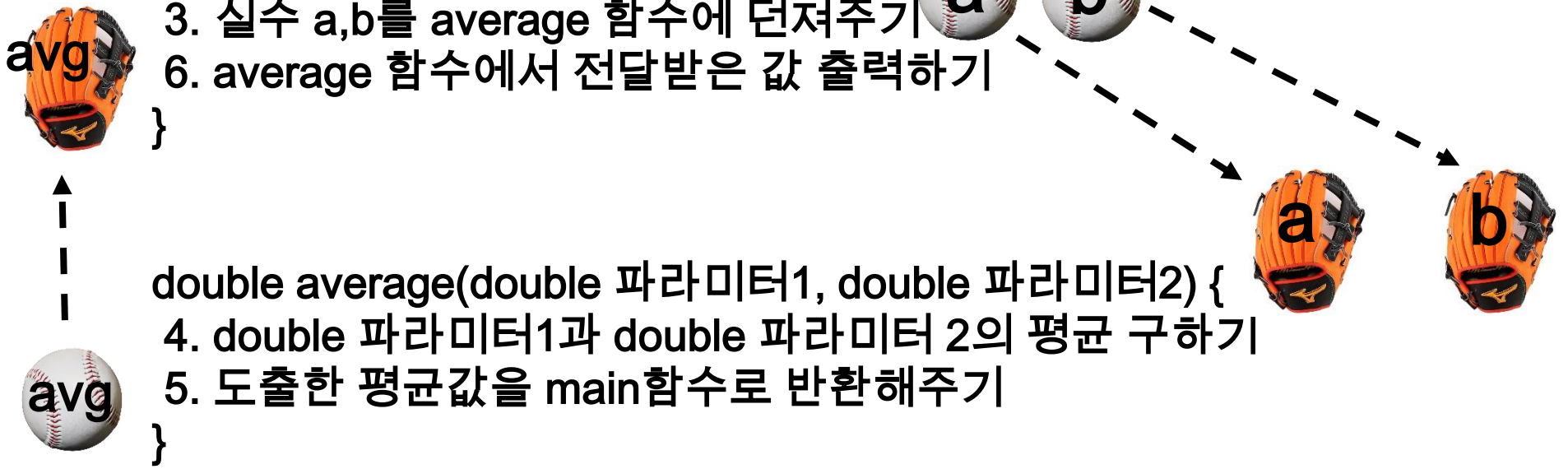
Q1

- 설계!!

```
#include<stdio.h>
double average (double 파라미터1, double 파라미터2);

int main() {
 1. 실수 a 입력받기
 2. 실수 b 입력받기
 3. 실수 a,b를 average 함수에 던져주기
 6. average 함수에서 전달받은 값 출력하기
}

double average(double 파라미터1, double 파라미터2) {
 4. double 파라미터1과 double 파라미터 2의 평균 구하기
 5. 도출한 평균값을 main함수로 반환해주기
}
```



Q1

- 코드

```c
#include<stdio.h>
double average(double arg_1, double arg_2);
double avg;

int main() {
    double num_1, num_2;

    printf("첫 번 째  실 수 를  입 력 하 세 요 : ");
    scanf("%lf", &num_1);
    printf("두 번 째  실 수 를  입 력 하 세 요 : ");
    scanf("%lf", &num_2);

    avg=average(num_1, num_2);

    printf("두  실 수 의  평 균 값 은  %lf 입 니 다 . \n", avg);
}

double average(double arg_1, double arg_2){
    avg = (arg_1+arg_2)/2;

    return(avg);
}
```

Q1

- 실행화면

```
첫 번 째  실 수 를  입 력 하 세 요 : 87.543
두 번 째  실 수 를  입 력 하 세 요 : 24.931
두  실 수 의  평 균 값 은  56.237000 입 니 다 .
```

- 문제의 포인트는!
  - 사용자 지정 함수를 잘 사용할 수 있는가
  - double형 데이터 타입을 잘 사용할 수 있는가

Q2

- 문제

2. 1부터 1000까지의 수를 출력하는데 있어서 다음의 조건을 만족하는 프로그램을 작성하시오.

   - 사용해야 하는 반복문: *for*
   - 한줄(한행)에 숫자 10개를 출력하고 행을 바꿔서 출력
   - 변수 $i = 1$ 을 직접 증가 시키지 않음
   - 변수 i을 지정하는 (가리키는) 포인터 변수 $*pi$ 을 사용해 증가

Q2

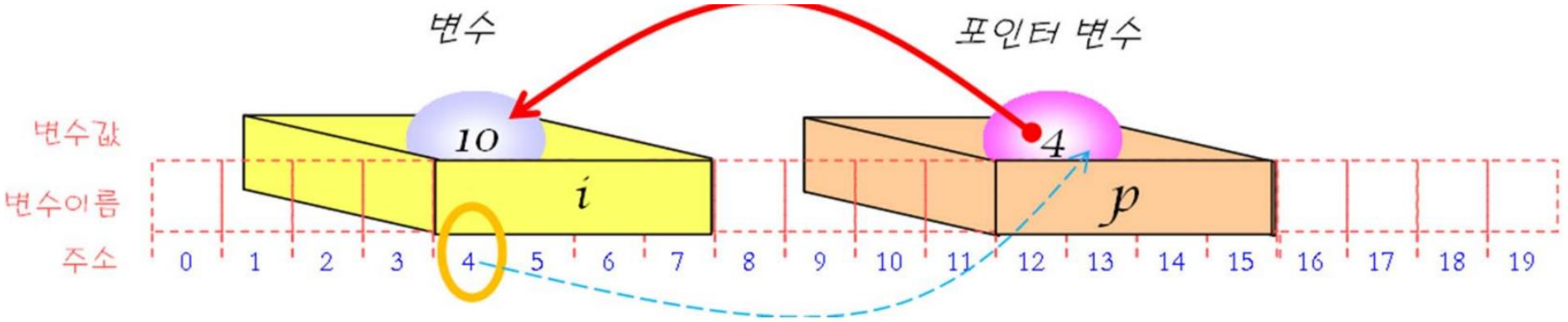
- 문제 분석!!

2. 1부터 1000까지의 수를 출력하는데 있어서 다음의 조건을 만족하는 프로그램을 작성하시오.

  - 사용해야 하는 반복문: *for* → for문을 사용해서 1부터 1000까지 출력!
  - 한줄(한행)에 숫자 10개를 출력하고 행을 바꿔서 출력 → 10 증가할 때 마다 "\n" 입력
  - 변수 $i=1$ 을 직접 증가 시키지 않음 → ....? (동공지진)
  - 변수 i을 지정하는 (가리키는) 포인터 변수 $*pi$ 을 사용해 증가 → 하... 포인터.....

Q2

- 포…인..터…?!
  - 다른 변수 혹은 그 변수의 메모리 주소값을 가리키는 변수 (32비트 기준 4바이트)



Q2

- 포…인..터…?!
  - 사용법

```
#include<stdio.h>

int main() {
    int i =300;
    int *p = &i;

    printf("&i=%u\n", &i);
    printf("i=%d\n", i);

    printf ("*p=%d\n", *p);
    printf("p=%u\n", p);
}
```

포인터 타입과 변수 타입은 일치해야해요!

```
&i=2434855564
i=300
*p=300
p=2434855564
```

//꼭 초기화를 해줘야 해요!

Q2

- 포…인..터…?!
  - 참조 값을 증가시키기 위해서는?
    - *p++ vs (*p)++
  - *p++
    - 값을 참조한 뒤 메모리 주소값을 증가시킴
  - (*p)++
    - 참조하는 값을 증가시킴

Q2

- 설계

```
#include<stdio.h>

int main() {
 1. 변수 및 포인터 변수 선언
 2. for문 (1000까지 출력)1
 3. 포인터 증가시키기
 4. 값 출력하기
 5. 10개 출력 시 개행 삽입
}
```

Q2

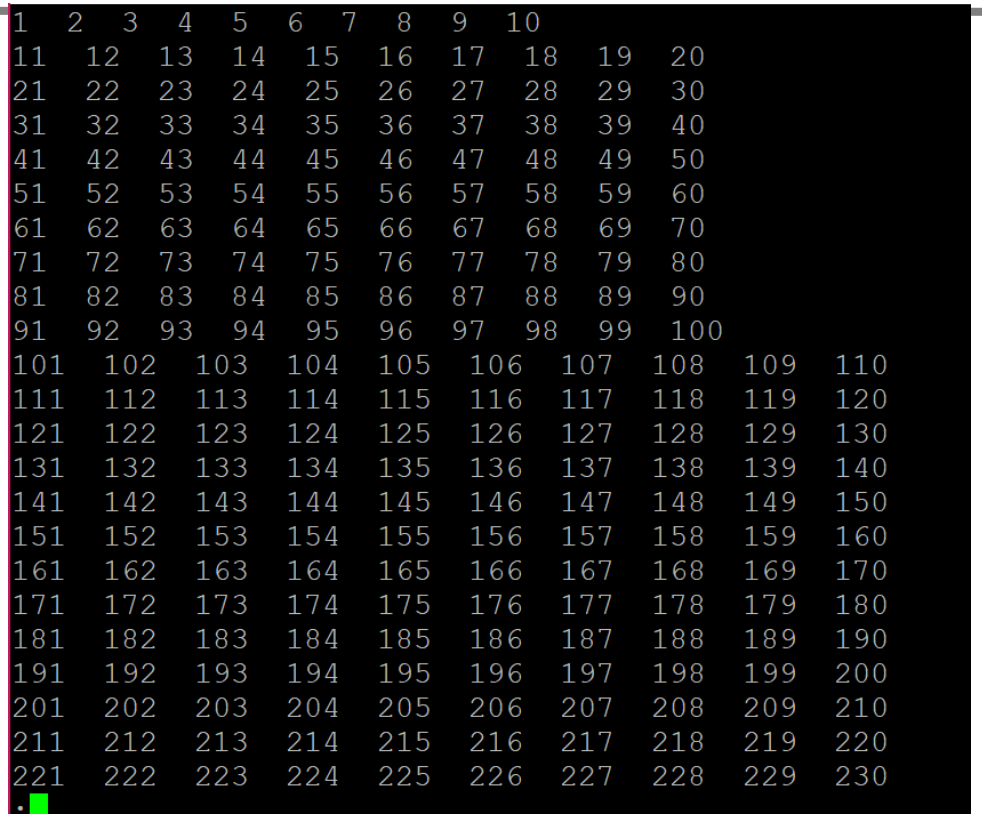
- 코드

```
 1 #include<stdio.h>
 2
 3 int main(){
 4       int i=0, j, k;
 5       int *pi=&i;
 6
 7       for (j=1; j<=1000; j++){
 8               (*pi)++;
 9               printf("%d  ", *pi);
10           if(j%10==0)
11               printf ("\n");
12       }
13 }
```

*pi 대신 i 넣어도 결과 같음

Q2

- 실행화면



- 문제의 포인트는!
  - 포인터를 잘 사용할 수 있는가 (포인터 변수 선언 및 참조 값 증감연산)
  - 반복문을 잘 사용할 수 있는가

Q3

- 문제

3. 포인터를 사용해 두 변수의 값을 교환하는 프로그램을 작성하시오.

    - 정의해야 하는 함수: *void swap(int *pa, int *pb)*

Q3

- 문제 분석!!

포인터를 선언해야겠군!

A, B → A, B

3. 포인터를 사용해 두 변수의 값을 교환하는 프로그램을 작성하시오.

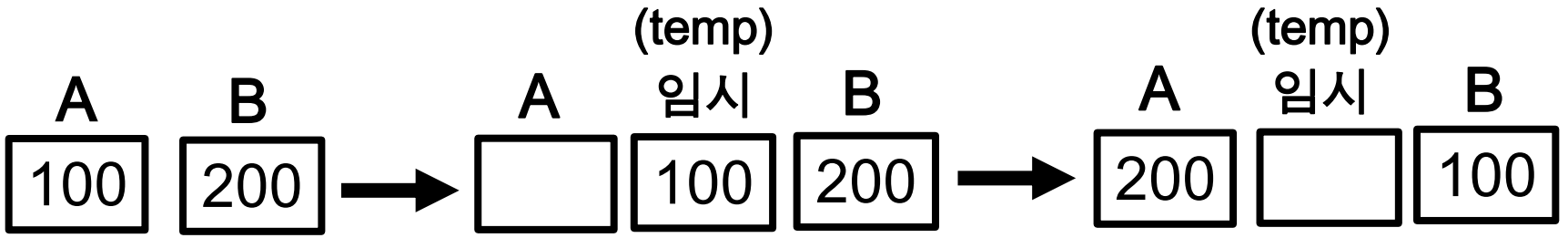
- 정의해야 하는 함수: *void swap(int *pa, int *pb)*

사용자 정의 함수에서 변수 값을 교환해야하는구나!

이것은!! 그 유명한 swap 함수?!!
입력받을 변수의 데이터 타입은 int형이군!
main함수에서 입력받아 아규먼트로 넘겨야겠네!!
void형 인걸보니 반환값은 없겠어!

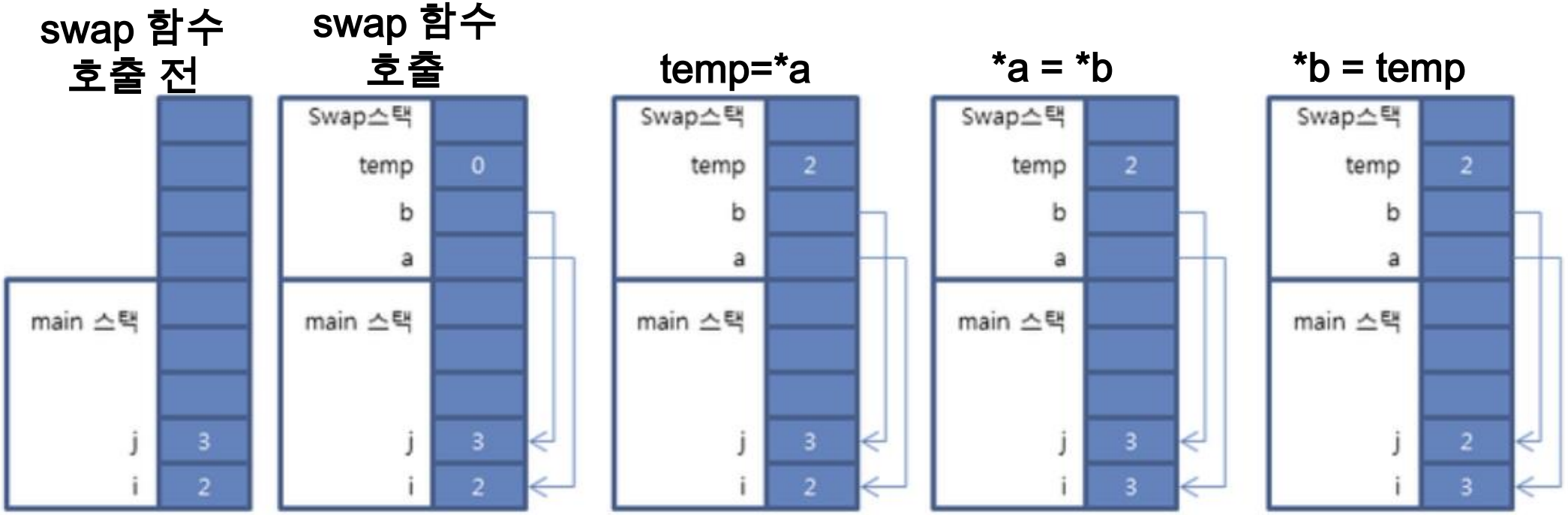
Q3

- swap 함수
  - 두 변수의 값을 서로 교환하여 바꾸는데 보편적으로 이용되는 유명한 함수!
    - temp 변수
      - A 변수에 100, B 변수에 200이 저장되어있을 때, 100을 B로, 200을 A로 옮기기 위해서는 임시 변수 필요
        - 일반적으로, 이 임시 변수를 temporary에서 딴 temp 변수라고 선언함



Q3

- 그런데.. 왜 void swap이죠..? main함수에 결과값을 반환해야하는거 아닌가요? 아예 swap 함수에서 출력하라는 건가요?
  - int *pa, int *pb
    - 포인터로 접근하기 때문에 반환하지 않아도 swap함수 종료와 동시에 이미 바뀌어있음



Q3

- 설계

```
#include<stdio.h>
void swap(int *pa, int *pb)

int main() {
 1. 두 정수 입력받기 (혹은 지정하기)
 2. swap 전 입력받은 두 변수 출력
 3. swap 함수에 아규먼트로 전달
 6. swap 함수 완료 후 두 변수 출력
}

void swap(int *pa, int *pb)
 4. temp 변수 선언
 5. swap
}
```

Q3

- 코드

```c
#include<stdio.h>
void swap(int *pa, int *pb);

int main(){
    int a,b;

    printf("a에 넣을 정수를 입력하세요 :");
    scanf("%d", &a);
    printf("b에 넣을 정수를 입력하세요 :");
    scanf("%d", &b);

    printf("\nswap 전 : a=%d, b=%d\n\n",a,b);

    swap(&a,&b);

    printf("swap 후 : a=%d, b=%d\n\n",a,b);
}


void swap(int *pa, int *pb){
    int temp;

    temp=*pa;
    *pa=*pb;
    *pb=temp;

    printf("swap진행 완료 (아직 swap함수 내부입니다): a=%d, b=%d\n\n", [redacted]);
}
```

QUIZ!!
아래 빈칸에는 무엇이 들어가야할까요??

Q3

- 실행화면

```
a에 넣을 정수를 입력하세요 :3
b에 넣을 정수를 입력하세요 :7

swap 전 : a=3, b=7

swap진행 완료 (아직 swap함수 내부입니다): a=7, b=3

swap 후 : a=7, b=3
```

- 문제의 포인트!!
  - swap함수를 잘 사용할 수 있는가
  - 사용자 정의 함수를 잘 사용할 수 있는가
  - 포인터를 잘 사용할 수 있는가

Q4

- 문제

4. 다음의 팩토리얼 계산 프로그램의 오류를 GDB 를 이용해 해결하고, 사용했던 GDB 명령/옵션을 소스코드에 주석으로 넣으시오.

```
# include <stdio.h>

int main()
{
        int i, num, j;
        printf ("Enter the number: ");
        scanf ("%d", &num );

        for (i=1; i<num; i++)
                j=j*i;

        printf("The factorial of %d is %d\n",num,j);
}
```

Q4

- 문제 분석!!

4. 다음의 팩토리얼 계산 프로그램의 오류를 GDB 를 이용해 해결하고, 사용했던 GDB 명령/옵션을 소스코드에 주석으로 넣으시오.

```
# include <stdio.h>

int main()
{
        int i, num, j;
        printf ("Enter the number: ");
        scanf ("%d", &num );

        for (i=1; i<num; i++)
                j=j*i;

        printf("The factorial of %d is %d\n",num,j);
}
```

의심포인트1
: num이 제대로 입력되었나

의심포인트2
: i가 제대로 입력되었나

의심포인트3
: j가 제대로 입력되었나

Q4

- gdb 명령어
  - 컴파일: gcc –g –o (실행파일명) (c파일명.c)
  - 코드 확인: list
  - 브레이크포인트 설정: b (라인넘버)
  - 브레이크포인트 전까지 실행: r
  - 다음 브레이크포인트까지 실행: c
  - 변수 확인: p (변수)

Q4

- 설계 및 주석 달기

```
1. #include<stdio.h>            //list로 코드 10줄 확인. b 7, b 9, b 10 이후 r로 프로그램 실행
2.
3.  int main() {
4.        int i, num, j;
5.  printf("Enter the number: ");
6.  scanf("%d", &num);      //num 변수 입력 확인을 위해 line 7에 브레이크포인트. b 7
                             //변수 i, num, j 확인. p i, p num, p j.
                              이후 c로 다음 브레이크포인트까지 진행
7.
8.        for(i=1; i<num; i++)   // i 변수 확인을 위해 line 9에 브레이크포인트. b 9
                                  //변수 i, num, j 확인. p i, p num, p j. 이후 c로 다음 브레이크
                                 포인트까지 진행
9.              j=j*1;            // i 변수 확인을 위해 line 9에 브레이크포인트. b 10
                                  //변수 i, num, j 확인. p i, p num, p j. 이후 c로 다음 브레이크
                                 포인트까지 진행
10.
11.       printf("The factorial of %d is %d\n", num, j);
12. }
```

Q4

- 실행화면

```
(gdb) list
1       #include<stdio.h>
2
3       int main(){
4           int i, num, j;
5           printf("Enter the number: ");
6           scanf("%d", &num);
7
8           for (i=1; i<num; i++)
9               j=j*i;
10
(gdb) b 7
Breakpoint 1 at 0x75a: file q4.c, line 7.
(gdb) b 9
Breakpoint 2 at 0x763: file q4.c, line 9.
(gdb) b 10
Breakpoint 3 at 0x779: file q4.c, line 10.
(gdb) r
```

Q4

- 실행화면

```
Breakpoint 2, main () at q4.c:9
9               j=j*i;
(gdb) p i
$4 = 1
(gdb) p num
$5 = 3
(gdb) p j
$6 = 32767
(gdb) c
Continuing.

Breakpoint 2, main () at q4.c:9
9               j=j*i;
(gdb) p i
$7 = 2
(gdb) p num
$8 = 3
(gdb) p j
$9 = 32767
(gdb) c
Continuing.

Breakpoint 3, main () at q4.c:11
11          printf("The factorial of %d is %d\n", num, j);
(gdb) p i
$10 = 3
(gdb) p j
$11 = 65534
(gdb) p num
```

감사합니다!

이미지 출처

- p6, p8
  - https://jnilbo.com/2019/02/11/20190211174407797 22/\
- p12
  - https://m.blog.naver.com/PostView.nhn?blogId=ansrud0995&logNo=220913308666&proxyReferer=https%3A%2F%2Fwww.google.com%2F
- p13
  - 생능출판사, 2012, 누구나 즐기는 c언어 콘서트, 제 9장 포인터
- p22
  - http://ehpub.co.kr/tag/%EB%91%90-%EA%B0%9C%EC%9D%98-%EB%B3%80%EC%88%98%EC%9D%98-%EA%B0%92%EC%9D%84-%EB%B0%94%EA%BE%B8%EA%B8%B0/